C interfaces to GALAHAD SILS

Jari Fowkes and Nick Gould
STFC Rutherford Appleton Laboratory
Thu Jun 22 2023

Chapter 1

GALAHAD C package sils

1.1 Introduction

1.1.1 Purpose

This package **solves sparse symmetric system of linear equations.**. Given an n by n sparse matrix $A = a_{ij}$, and an n vector b, the package solves the system $Ax = b$. The matrix A need not be definite. There is an option for iterative refinement.

Currently, only the control and inform parameters are exposed; these are provided and used by other GALAHAD packages with C interfaces. Extended functionality is available using the GALAHAD package sls.

1.1.2 Authors

N. I. M. Gould, STFC-Rutherford Appleton Laboratory, England.

C interface, additionally J. Fowkes, STFC-Rutherford Appleton Laboratory.

Julia interface, additionally A. Montoison and D. Orban, Polytechnique Montréal.

1.1.3 Originally released

April 2001, C interface December 2021.

1.1.4 Method

The method used is a direct method based on a sparse variant of Gaussian elimination and is discussed further by

I. S. Duff and J. K. Reid (1983), ACM Trans. Math. Software **9** pp.302-325.

1.1.5 Symmetric matrix storage formats

The symmetric n by n coefficient matrix A may be presented and stored in a variety of convenient input formats. Crucially symmetry is exploited by only storing values from the lower triangular part (i.e, those entries that lie on or below the leading diagonal).

Both C-style (0 based) and fortran-style (1-based) indexing is allowed. Choose `control.f_indexing` as `false` for C style and `true` for fortran style; the discussion below presumes C style, but add 1 to indices for the corresponding fortran version.

Wrappers will automatically convert between 0-based (C) and 1-based (fortran) array indexing, so may be used transparently from C. This conversion involves both time and memory overheads that may be avoided by supplying data that is already stored using 1-based indexing.

1.1.5.1 Dense storage format

The matrix A is stored as a compact dense matrix by rows, that is, the values of the entries of each row in turn are stored in order within an appropriate real one-dimensional array. Since A is symmetric, only the lower triangular part (that is the part A_{ij} for $0 \leq j \leq i \leq n-1$) need be held. In this case the lower triangle should be stored by rows, that is component $i * i/2 + j$ of the storage array val will hold the value A_{ij} (and, by symmetry, A_{ji}) for $0 \leq j \leq i \leq n-1$.

1.1.5.2 Sparse co-ordinate storage format

Only the nonzero entries of the matrices are stored. For the l-th entry, $0 \leq l \leq ne-1$, of A, its row index i, column index j and value A_{ij}, $0 \leq j \leq i \leq n-1$, are stored as the l-th components of the integer arrays row and col and real array val, respectively, while the number of nonzeros is recorded as ne = ne. Note that only the entries in the lower triangle should be stored.

1.1.5.3 Sparse row-wise storage format

Again only the nonzero entries are stored, but this time they are ordered so that those in row i appear directly before those in row i+1. For the i-th row of A the i-th component of the integer array ptr holds the position of the first entry in this row, while ptr(n) holds the total number of entries. The column indices j, $0 \leq j \leq i$, and values A_{ij} of the entries in the i-th row are stored in components l = ptr(i), . . ., ptr(i+1)-1 of the integer array col, and real array val, respectively. Note that as before only the entries in the lower triangle should be stored. For sparse matrices, this scheme almost always requires less storage than its predecessor.

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

File Documentation

3.1 galahad_sils.h File Reference

```
#include <stdbool.h>
#include <stdint.h>
#include "galahad_precision.h"
#include "galahad_cfunctions.h"
```

Data Structures

- struct sils_control_type
- struct sils_ainfo_type
- struct sils_finfo_type
- struct sils_sinfo_type

Functions

- void sils_initialize (void **data, struct sils_control_type *control, int *status)
- void sils_read_specfile (struct sils_control_type *control, const char specfile[ ])
- void sils_import (struct sils_control_type *control, void **data, int *status)
- void sils_reset_control (struct sils_control_type *control, void **data, int *status)
- void sils_information (void **data, struct sils_ainfo_type *ainfo, struct sils_finfo_type *finfo, struct sils_sinfo_type *sinfo, int *status)
- void sils_finalize (void **data, struct sils_control_type *control, int *status)

3.1.1 Data Structure Documentation

3.1.1.1 struct sils_control_type

control derived type as a C struct

Data Fields

bool	f_indexing	use C or Fortran sparse matrix indexing
int	ICNTL[30]	MA27 internal integer controls.
int	lp	Unit for error messages.
int	wp	Unit for warning messages.
int	mp	Unit for monitor output.
int	sp	Unit for statistical output.
int	ldiag	Controls level of diagnostic output.
int	la	Initial size for real array for the factors. If less than nrlnec, default size used.
int	liw	Initial size for integer array for the factors. If less than nirnec, default size used.
int	maxla	Max. size for real array for the factors.
int	maxliw	Max. size for integer array for the factors.
int	pivoting	Controls pivoting. Possible values are: • 1 Numerical pivoting will be performed. • 2 No pivoting will be performed and an error exit will occur immediately a pivot sign change is detected. • 3 No pivoting will be performed and an error exit will occur if a zero pivot is detected. • 4 No pivoting is performed but pivots are changed to all be positive.
int	nemin	Minimum number of eliminations in a step (unused)
int	factorblocking	Level 3 blocking in factorize (unused)
int	solveblocking	Level 2 and 3 blocking in solve.
int	thresh	Controls threshold for detecting full rows in analyse, registered as percentage of N, 100 Only fully dense rows detected (default)
int	ordering	Controls ordering: Possible values are: • 0 AMD using HSL's MC47 • 1 User defined • 2 AMD using HSL's MC50 • 3 Min deg as in HSL's MA57 • 4 Metis_nodend ordering • 5 Ordering chosen depending on matrix characteristics. At the moment choices are HSL's MC50 or Metis_nodend • $>$5 Presently equivalent to 5 but may chnage
int	scaling	Controls scaling: Possible values are: • 0 No scaling • $>$0 Scaling using HSL's MC64 but may change for $>$ 1
real_wp_	CNTL[5]	MA27 internal real controls.
real_wp_	multiplier	Factor by which arrays sizes are to be increased if they are too small.
real_wp_	reduce	If previously allocated internal workspace arrays are greater than reduce times the currently required sizes, they are reset to current requirment.
real_wp_	u	Pivot threshold.
real_wp_	static_tolerance	used for setting static pivot level

Data Fields

real_wp_	static_level	used for switch to static
real_wp_	tolerance	Anything less than this is considered zero.
real_wp_	convergence	used to monitor convergence in iterative refinement

3.1.1.2 struct sils_ainfo_type

ainfo derived type as a C struct

Data Fields

int	flag	Flags success or failure case.
int	more	More information on failure.
int	nsteps	Number of elimination steps.
int	nrltot	Size for a without compression.
int	nirtot	Size for iw without compression.
int	nrlnec	Size for a with compression.
int	nirnec	Size for iw with compression.
int	nrladu	Number of reals to hold factors.
int	niradu	Number of integers to hold factors.
int	ncmpa	Number of compresses.
int	oor	Number of indices out-of-range.
int	dup	Number of duplicates.
int	maxfrt	Forecast maximum front size.
int	stat	STAT value after allocate failure.
int	faulty	legacy component, now not used
real_wp_	opsa	Anticipated number of operations in assembly.
real_wp_	opse	Anticipated number of operations in elimination.

3.1.1.3 struct sils_finfo_type

finfo derived type as a C struct

Data Fields

int	flag	Flags success or failure case.
int	more	More information on failure.
int	maxfrt	Largest front size.
int	nebdu	Number of entries in factors.
int	nrlbdu	Number of reals that hold factors.
int	nirbdu	Number of integers that hold factors.
int	nrltot	Size for a without compression.
int	nirtot	Size for iw without compression.
int	nrlnec	Size for a with compression.
int	nirnec	Size for iw with compression.
int	ncmpbr	Number of compresses of real data.

Data Fields

int	ncmpbi	Number of compresses of integer data.
int	ntwo	Number of 2x2 pivots.
int	neig	Number of negative eigenvalues.
int	delay	Number of delayed pivots (total)
int	signc	Number of pivot sign changes when control.pivoting=3.
int	nstatic	Number of static pivots chosen.
int	modstep	First pivot modification when control.pivoting=4.
int	rank	Rank of original factorization.
int	stat	STAT value after allocate failure.
int	faulty	legacy component, now not used
int	step	legacy component, now not used
real_wp_	opsa	### 3.1.2 operations in assembly
real_wp_	opse	number of operations in elimination
real_wp_	opsb	Additional number of operations for BLAS.
real_wp_	maxchange	Largest control.pivoting=4 modification.
real_wp_	smin	Minimum scaling factor.
real_wp_	smax	Maximum scaling factor.

3.1.2.1 struct sils_sinfo_type

sinfo derived type as a C struct

Data Fields

int	flag	Flags success or failure case.
int	stat	STAT value after allocate failure.
real_wp_	cond	Condition number of matrix (category 1 eqs)
real_wp_	cond2	Condition number of matrix (category 2 eqs)
real_wp_	berr	Backward error for the system (category 1 eqs)
real_wp_	berr2	Backward error for the system (category 2 eqs)
real_wp_	error	Estimate of forward error.

3.1.3 Function Documentation

3.1.3.1 sils_initialize()

```
void sils_initialize (
             void ** data,
```

```
            struct sils_control_type * control,
            int * status )
```

Set default control values and initialize private data

Parameters

in,out	*data*	holds private internal data
out	*control*	is a struct containing control information (see sils_control_type)
out	*status*	is a scalar variable of type int, that gives the exit status from the package. Possible values are (currently): • 0. The values were recorded succesfully

3.1.3.2 sils_read_specfile()

```
void sils_read_specfile (
            struct sils_control_type * control,
            const char specfile[] )
```

Read the content of a specification file, and assign values associated with given keywords to the corresponding control parameters. By default, the spcification file will be named RUNSILS.SPC and lie in the current directory. Refer to Table 2.1 in the fortran documentation provided in $GALAHAD/doc/sils.pdf for a list of keywords that may be set.

Parameters

in,out	*control*	is a struct containing control information (see sils_control_type)
in	*specfile*	is a character string containing the name of the specification file

3.1.3.3 sils_import()

```
void sils_import (
            struct sils_control_type * control,
            void ** data,
            int * status )
```

Import problem data into internal storage prior to solution.

Parameters

in	*control*	is a struct whose members provide control paramters for the remaining prcedures (see sils_control_type)
in,out	*data*	holds private internal data

Parameters

in,out	*status*	is a scalar variable of type int, that gives the exit status from the package. Possible values are: • 1. The import was succesful, and the package is ready for the solve phase • -1. An allocation error occurred. A message indicating the offending array is written on unit control.error, and the returned allocation status and a string containing the name of the offending array are held in inform.alloc_status and inform.bad_alloc respectively. • -2. A deallocation error occurred. A message indicating the offending array is written on unit control.error and the returned allocation status and a string containing the name of the offending array are held in inform.alloc_status and inform.bad_alloc respectively. • -3. The restriction n $>$ 0 or requirement that type contains its relevant string 'dense', 'coordinate', 'sparse_by_rows', 'diagonal' or 'absent' has been violated.

3.1.3.4 sils_reset_control()

```
void sils_reset_control (
             struct sils_control_type * control,
             void ** data,
             int * status )
```

Reset control parameters after import if required.

Parameters

in	*control*	is a struct whose members provide control paramters for the remaining prcedures (see sils_control_type)
in,out	*data*	holds private internal data
in,out	*status*	is a scalar variable of type int, that gives the exit status from the package. Possible values are: • 1. The import was succesful, and the package is ready for the solve phase

3.1.3.5 sils_information()

```
void sils_information (
             void ** data,
             struct sils_ainfo_type * ainfo,
             struct sils_finfo_type * finfo,
             struct sils_sinfo_type * sinfo,
             int * status )
```

Provides output information

Parameters

in,out	*data*	holds private internal data
out	*ainfo*	is a struct containing output information (see sils_ainfo_type)
out	*finfo*	is a struct containing output information (see sils_finfo_type)
out	*sinfo*	is a struct containing output information (see sils_sinfo_type)
out	*status*	is a scalar variable of type int, that gives the exit status from the package. Possible values are (currently): • 0. The values were recorded succesfully

3.1.3.6 sils_finalize()

```
void sils_finalize (
            void ** data,
            struct sils_control_type * control,
            int * status )
```

Deallocate all internal private storage

Parameters

in,out	*data*	holds private internal data
out	*control*	is a struct containing control information (see sils_control_type)
out	*status*	is a scalar variable of type int, that gives the exit status from the package. Possible values are (currently): • 0. The values were recorded succesfully • $\neq$ 0. The Fortran STAT value of an allocate or deallocate statement that has failed.

Chapter 4

Example Documentation

4.1 silst.c

This is an example of how to use the package.